AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. – 25. (Cancelled)

26. (Currently Amended) A method for handling sharing of a physical memory space between a first process and a second process, the method comprising:

allocating a portion of the physical memory space and creating a buffer object responsive to a buffer object request~~a first address range in the first process and a second address range in the second process~~, wherein one of the processes executes native code of a program and the other process executes safe language code of the program; and

mapping [[the]] a first address range of the first process and a [[the]] second address range of the second process to the allocated portion of the shared physical memory space, wherein the buffer object represents at least one of the address ranges.

27. (Currently Amended) The method of claim 26, wherein at least one of the ~~first and second~~ address ranges comprises virtual addresses.

28. (Previously Presented) The method of claim 26, wherein the physical memory space comprises a set of one or more physical pages.

29. (Currently Amended) The method of claim 26, further comprising the second process creating the buffer object and indicating the allocated physical memory space portion and the buffer object to the first process~~wherein the allocating is responsive to the first process requesting a direct buffer for the first address range from the second process~~.

30. (Currently Amended) The method of claim 29, wherein the safe language code comprises Java language code ~~further comprising the second process:~~

~~causing generation of a direct buffer object and an associated direct buffer object identifier; and~~

~~communicating the direct buffer object identifier and a physical memory space identifier for the shared physical memory space to the first process~~.

31. (Currently Amended) The method of claim 29 further comprising the first process indicating ~~communicating~~ the ~~direct~~ buffer object ~~identifier and the physical memory space identifier~~ to one or more callers~~other processes that execute native code~~.

32. (Currently Amended) The method of claim 26 further comprising maintaining an encoding that indicates allocated portions of the physical memory space to allow detection of at least one of overlapping address ranges and nested address ranges ~~within one of the first and second processes~~.

33. – 35. (Cancelled)

36. (Currently Amended) The method of claim 50, [[35]] further comprising ~~the first process~~ communicating an identifier of the created direct buffer ~~object identifier to a native code~~ to one or more callers.

37. (Currently Amended) The method of claim 50, [[33]] wherein the memory space ~~physical address ranges~~ comprises a set of one or more physical memory pages.

38. (Currently Amended) The method of claim 50, [[33]] further comprising allowing ~~at least two of the~~ address ranges in the first process to overlap and/or nest.

39. (Previously Presented) The method of claim 38 further comprising:

the first process requesting an address range [A2, A2+S2] that overlaps with a previously allocated address range [A1, A1+S1], wherein A1 and A2 represent addresses in the first process and S1 and S2 represent memory space sizes;

the second process,

allocating an address range [A2', A2'+S2], wherein A2' represent addresses in the second process,

mapping [A2', A2' + (A1+S1-A2)] in the second process to a same first portion of the one or more physical memory spaces to which [A2, A1+S1] in the first process is mapped, and

mapping [A2'+(A1+S1-A2), A2'+S2] in the second process to a same second portion of the one or more physical memory spaces to which [A1+S1, A2+S2] in the first process is mapped,

wherein A1 < A2 < A1+S1 < A2+S2.

40. (Currently Amended) The method of claim 50 [[33]] further comprising maintaining a list that indicates mapped [[the]] address ranges to allow~~, wherein the list allows~~ detection of at least one of overlapping address ranges and nested address ranges.

41. (Currently Amended) The method of claim 50, [[33]], wherein at least one of the first process address range[[s]] and the second process address range[[s]] comprise virtual addresses.

42. (Currently Amended) A computer program encoded on one or more computer readable media, the computer program comprising:

a first language code executable to allocate an address range in a first environment, which executes the first language code, responsive to a request for a buffer object, ~~and executable~~ to map the first environment address range to a portion of a ~~physical~~ memory space, to create the buffer object as representative of the allocated address range, and to communicate the buffer object and the portion of memory space to another environment that executes a different language code; and

a second language code executable to request the buffer object and to allocate an address range in a second environment, which executes the second language code, and executable to map the second environment address range to the portion of the ~~physical~~ memory space ~~responsive to indication of the physical memory space from the first language code~~,

~~wherein the first environment address range is correspondent to the second environment address range~~,

wherein one of the first and second language codes comprises a safe language code and the other of the first and second language codes comprises a native code.

43. (Previously Presented) The computer program encoding of claim 42 further comprising an interface code to handle communications between the first and second environments.

44. (Previously Presented) The computer program encoding of claim 43, wherein the interface code is implemented according to the Java native interface, the safe language code is implemented according to the Java programming language, and the one of the first and second environments that executes the safe language code comprises a virtual machine.

45. (Previously Presented) The computer program encoding of claim 44, wherein the other of the first and second environments that executes native code comprises an operating system.

46. (Cancelled)

47. (Currently Amended) An apparatus comprising:

a memory; and

means for mapping ~~allocating~~ an address range in a first process for a first code and an ~~analogous~~ address range in a second process for a second code to a same region of the memory to facilitate transparent code isolation, and for representing at least one of the address ranges with a buffer object~~sharing of at least a portion of the memory between the first code and the second code~~,

wherein one of the first and second processes executes native code and the other of the first and second processes executes safe language code.

48. (Currently Amended) The apparatus of claim 47 further comprising means for communicating an identifier of the buffer object between the first and second processes~~, wherein~~

~~at least one of the first process address range and the second process address range comprises virtual addresses~~.

49. (Currently Amended) The apparatus of claim 47 further comprising means for detecting~~allowing detection of~~ at least one of nested address ranges and overlapping address ranges.

50. (New) A method of performing native code isolation in the presence of direct buffers without losing transparency, the method comprising:

mapping a first address range of a first process and a second address range of a second process to a same portion of memory space, wherein one of the first and second processes executes native code and the other of the first and second processes executes a safe language code; and

creating a direct buffer to represent at least one of the first and the second address ranges.

51. (New) The method of claim 50, wherein the direct buffer is created by the second process responsive to receiving a direct buffer request from the first process.

52. (New) The method of claim 50, wherein the same portion of memory space is allocated from one of a memory mapped file and a frame buffer.

53. (New) The computer program product encoding of claim 42, wherein the second language code is further executable to indicate the buffer object to a caller.